Spanning Trees with Many Leaves in Graphs without Diamonds and Blossoms

Paul Bonsma[*]   Florian Zickfeld [†]

Technische Universität Berlin, Fachbereich Mathematik
Str. des 17. Juni 136, 10623 Berlin, Germany
{bonsma,zickfeld}@math.tu-berlin.de

Abstract

It is known that graphs on n vertices with minimum degree at least 3 have spanning trees with at least $n/4 + 2$ leaves and that this can be improved to $(n + 4)/3$ for cubic graphs without the diamond $K_4 - e$ as a subgraph. We generalize the second result by proving that every graph with minimum degree at least 3, without diamonds and certain subgraphs called blossoms, has a spanning tree with at least $(n+4)/3$ leaves, and generalize this further by allowing vertices of lower degree. We show that it is necessary to exclude blossoms in order to obtain a bound of the form $n/3 + c$.

We use the new bound to obtain a simple FPT algorithm, which decides in $O(m) + O^*(6.75^k)$ time whether a graph of size m has a spanning tree with at least k leaves. This improves the best known time complexity for MAX LEAF SPANNING TREE.

1   Introduction

In this paper we study spanning trees with many leaves. We prove a new extremal result, and apply it to obtain a fast FPT algorithm for the related decision problem MaxLeaf.

We first introduce the extremal problem and explain our contribution. Throughout this paper G is assumed to be a simple and connected graph on $n \geq 2$ vertices. Other graphs may be multi-graphs, disconnected, or a K_1. The minimum vertex degree of G is denoted by $\delta(G)$. Vertices of degree 1 are called leaves.

Linial and Sturtevant [12] and Kleitman and West [11] showed that every graph G with $\delta(G) \geq 3$ has a spanning tree with at least $n/4+2$ leaves, and that this bound is best possible. The paper [11] also improves on this bound for graphs of higher minimum degree.

The examples showing that $n/4 + 2$ is best possible for graphs of minimum degree 3 all consist of cubic diamonds connected in a cyclic manner. A *diamond* is the graph K_4 minus one edge, and an induced diamond subgraph of a graph G is a *cubic diamond* if its four vertices all have degree 3 in G, see Figure 1 (a).

Since these examples are very restricted it is natural to ask if better bounds can be obtained when diamonds are forbidden as subgraphs. This question was answered by Griggs, Kleitman and Shastri [10] for *cubic* graphs, which are graphs where every vertex has degree 3.
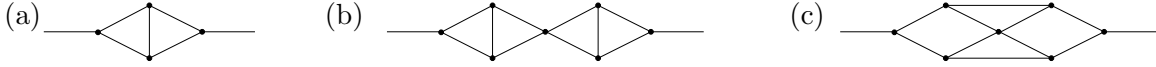
Figure 1: A cubic diamond (a), a 2-necklace (b), and a 2-blossom (c).

They show that a cubic graph G without diamonds always admits a spanning tree with at least $n/3 + 4/3$ leaves. For minimum degree 3 the following bound is proved in [2]. A graph G with $\delta(G) \geq 3$, without cubic diamonds, contains a spanning tree with at least $2n/7 + 12/7$ leaves. Both bounds are best possible for their respective classes.

In [2] it is conjectured that the following statement holds, which would improve the bound $2n/7 + 12/7$ with only a minor extra restriction, and would also generalize the result for cubic graphs from [10]: Every graph G with $\delta(G) \geq 3$ and without *2-necklaces* contains a spanning tree with at least $n/3 + 4/3$ leaves. Informally speaking, a 2-necklace is a concatenation of $k \geq 1$ diamonds with only two outgoing edges, see Figure 1 (b).

In Section 2 we disprove this conjecture by constructing graphs with $\delta(G) = 3$ without 2-necklaces, which do not admit spanning trees with more than $4n/13 + 2$ leaves. On the positive side, we prove that the statement is true after only excluding one more very specific structure, called a *2-blossom*, see Figure 1 (c). Precise definitions of 2-necklaces and 2-blossoms are given in Section 2. So we prove that graphs G with $\delta(G) \geq 3$ without 2-necklaces or 2-blossoms have a spanning tree with at least $n/3 + 4/3$ leaves. In fact we generalize this statement even further by removing any restriction on the minimum degree. The resulting statement is given in Theorem 1, which is our main result.

Let $V_{\geq 3}(G)$ denote the set of vertices in G with degree at least 3 and $n_{\geq 3}(G)$ its cardinality. Let $\ell(T)$ be the number of leaves of a graph T.

$V_{\geq 3}(G)$
$n_{\geq 3}(G)$
$\ell(T)$

Theorem 1. *Let G be a simple, connected graph on at least two vertices which contains neither 2-necklaces nor 2-blossoms. Then, G has a spanning tree T with*

$$\ell(T) \geq n_{\geq 3}(G)/3 + \begin{cases} 4/3 & \text{if } \delta(G) \geq 3 \\ 2 & \text{if } \delta(G) \leq 2. \end{cases}$$

Section 2 shows that Theorem 1 is also best possible for non-cubic graphs. Without proof we remark that also other results can be generalized in a similar way, e.g. it is not hard to extend the proof in [11] to prove that all graphs G have a spanning tree with at least $(n - n_2(G))/4 + c$ leaves, where $n_2(G)$ denotes the number of vertices of degree 2 in G.

Our proof of Theorem 1 is constructive and can be turned into a polynomial time algorithm for the construction of a spanning tree. The main technical contribution of this paper is that we prove this generalization of the statement in [10], and improvement of the statement in [2], without a proof as lengthy as the proofs in these two papers. This is made possible by extending the techniques and proofs from [10]. In Section 3 we argue that the long case study in [10] actually proves a strong new lemma, which we use as an important step in the proof of Theorem 1. We share the opinion expressed in [10] that a shorter proof of the bound for cubic graphs might not exist. Therefore using that result in order to prove the more general statement seems appropriate.

We now explain the consequences that Theorem 1 has for FPT algorithms (short for *fixed parameter tractable*) for the following decision problem.

2

Max-Leaves Spanning Tree (MaxLeaf):
INSTANCE: A graph G and integer k.
QUESTION: Does G have a spanning tree T with $\ell(T) \geq k$?

It is known that MaxLeaf is $\mathcal{NP}$-complete, see [9]. When choosing k as a parameter, an algorithm for this problem is called an *FPT algorithm* if its complexity is bounded by $f(k)g(n)$, where $g(n)$ is a polynomial. See [8] and [5] for introductions to FPT algorithms. $f(k)$ is called the *parameter function* of the algorithm. Usually, $g(n)$ will turn out to be a low degree polynomial, thus to assess the speed of the algorithm it is mainly important to consider the growth rate of $f(k)$. Though since MaxLeaf is $\mathcal{NP}$-complete, $f(k)$ will most likely always be exponential. Bodlaender [1] constructed the first FPT algorithm for MaxLeaf with a parameter function of roughly $(17k^4)!$. Since then, considerable effort has been put in finding faster FPT algorithms for this problem, see e.g. [4, 7, 3, 6, 2]. The papers [3, 6, 2] also establish a strong connection between extremal graph-theoretic results and fast FPT algorithms. In [3], the bound of $n/4 + 2$ from [11] mentioned above is used to find an FPT algorithm with parameter function $O^*(\binom{4k}{k}) \subset O^*(9.49^k)$. Here the O^* notation ignores polynomial factors. With the same techniques the bound of $2n/7 + 12/7$ mentioned above is is turned into the so far fastest algorithm, with a parameter function in $O^*(\binom{3.5k}{k}) \subset O^*(8.12^k)$ ([2]). Similarly Theorem 1 yields a new FPT algorithm for MaxLeaf, presented in Section 4.

FPT algorithm

parameter function

O^*

Theorem 2. *There exists an FPT algorithm for* MaxLeaf *with time complexity* $O(m) + O^*(6.75^k)$, *where* m *denotes the size of the input graph and* k *the desired number of leaves.*

This algorithm is the fastest FPT algorithm for MaxLeaf at the moment, both optimizing the dependency on the input size and the parameter function. It simplifies the ideas introduced by Bonsma, Brueggemann and Woeginger [3] and is also significantly simpler than the other recent fast FPT algorithms. Hardly any preprocessing of the input graph is needed, since Theorem 1 is already formulated for a very broad graph class. We end in Section 5 with a discussion of possible extensions and further consequences of Theorem 1.

2   Obstructions for Spanning Trees with Many Leaves

2.1   Diamond Necklaces, Blossoms and Flowers

As mentioned in the introduction 2-necklaces have been identified as an obstruction for the existence of spanning trees with $n/3 + c$ leaves in graphs with minimum degree 3, see [11] and [2]. In this section we show that they are not the only such obstruction. We start by precisely defining 2-necklaces and 2-blossoms.

The degree of a vertex v in a graph G is denoted by $d_G(v)$ and by $d(v)$ if ambiguities can be excluded. A vertex v of a subgraph H of G with $d_H(v) < d_G(v)$ is called a *terminal* of H.

terminal

Definition 1 (2-Necklace). The graph K_4 minus one edge is called a *diamond* and denoted by N_1. The degree 3 vertices are the *inner vertices* of the diamond.

For $k \geq 2$ the *diamond necklace* N_k is obtained from the graph N_{k-1} and a vertex disjoint N_1 by identifying a degree 2 vertex of N_1 with a degree 2 vertex of N_{k-1}. Thus, N_k has again two degree 2 vertices, which are denoted by c_1 and c_2.

An N_k subgraph of G is a *2-necklace* if it only has c_1 and c_2 as terminals, which both have degree 3 in G. See Figures 1 (a) and (b). If G contains an N_1 this way, this N_1 subgraph is also called a *cubic diamond* of G.

diamond

inner vertices

diamond necklace

2-necklace

cubic diamond

3

Diamond necklaces will also be called *necklaces* for short. In the course of studying the leafy tree problem we found that the subgraphs defined next are also an obstacle for the existence of spanning trees with $n/3 + c$ leaves.

Definition 2 (2-Blossom)**.** The graph B on seven vertices shown in Figure 2 (a) is the blossom graph. A blossom subgraph B of G is a 2-blossom if c_1 and c_2 are its only terminals, and they both have degree 3 in G, see Figure 2 (b).
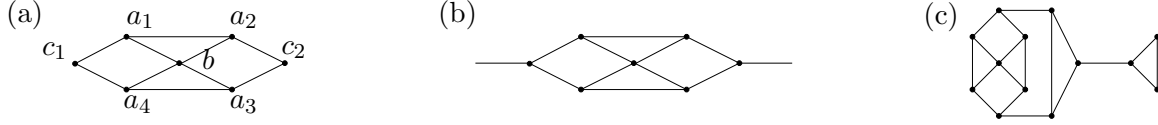

Figure 2: A blossom graph (a), a 2-blossom (b), and a flower (c).

If G contains a 2-blossom B, only the vertex b has degree 4 in G, and the remaining vertices of B have degree 3 in G. The two outgoing edges of a 2-necklace respectively a 2-blossom may in fact be the same edge, in that case G is just a 2-necklace respectively 2-blossom plus one additional edge. The next proposition shows how many leaves can be gained within a blossom.
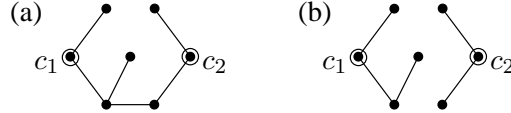

Figure 3: Spanning trees restricted to a blossom

Proposition 1. *Let G be a graph with a blossom subgraph B that has c_1 and c_2 as its only terminals. A spanning tree T of G exists with maximum number of leaves, such that $E(T) \cap E(B)$ has one of the forms in Figure 3.*

Proof. Consider a spanning tree T of G with maximum number of leaves. We may distinguish the following two cases for $E(T) \cap E(B)$: either this edge set induces a tree, or it induces a forest with two components, one containing c_1 and the other containing c_2.

In the first case, at most three non-terminal vertices of B can be leaves of T, since a path from c_1 to c_2 contains at least two internal vertices. In addition, if one of c_1 and c_2 is a leaf of T, then T can be seen to have at most two non-terminal vertices of B among its leaves. Since c_1 and c_2 together form a vertex cut of G, one of them is not a leaf in T. It follows that replacing $E(T) \cap E(B)$ by the edge set in Figure 3 (a) does not decrease the number of leaves. Since this edge set forms again a spanning tree of B, the resulting graph is a spanning tree of G.

Now suppose $E(T) \cap E(B)$ forms two components. At most four non-terminal vertices of B can be leaves of T. If one of c_1 and c_2 is a leaf of T, then T can have at most three non-terminal vertices of B among its leaves. One of c_1 and c_2 is not a leaf in T, and thus it follows again that replacing $E(T) \cap E(B)$ by the edge set in Figure 3 (b) does not decrease the number of leaves, while maintaining a spanning tree of G. $\qquad\square$

We now present a family of graphs with minimum degree 3 which do not contain diamond necklaces but do not have spanning trees with $n/3 + c$ leaves.

Definition 3 (Flower, Flowerbed)**.** The *flower graph* is the graph on thirteen vertices shown in Figure 2 (c).

The *flowerbed* R_i of length i consists of i flowers, connected in a cyclic manner, see Figure 4. Formally, R_i is constructed by starting with i disjoint flowers, and adding i edges in such a way that the graph is connected and has minimum degree 3.

Figure 4 shows the flowerbed R_5. The solid edges show a spanning tree with $4n/13 + 2$ leaves, which we will show to be optimal.
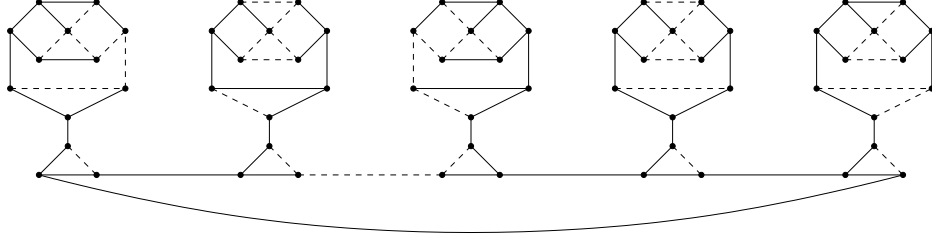

Figure 4: The flowerbed R_5. The solid edges show a tree with maximum number of leaves.

Proposition 2. *The flowerbed R_i has no spanning tree with more than $4n/13 + 2$ leaves.*

Proof. Let F be a flower in R_i containing a blossom B, where c_1 and c_2 are the terminals of B. The neighbor of c_j not in B is called f_j ($j = 1, 2$). We will argue that no spanning tree T of R_i has more than four leaves among $V(B) \cup \{f_1, f_2\}$. Proposition 1 shows that without loss of generality we may assume that $E(T) \cap E(B)$ has one of the two forms in Figure 3. If it has the first form, then one of f_1 and f_2 may be a leaf of T, but not both since together they form a vertex cut of R_i. If $E(T) \cap E(B)$ has the second form, then f_1 and f_2 are both cut vertices of T, so neither can be a leaf.

Now we consider the other vertices of R_i that may be leaves in T. Let C be the cycle in R_i that joins the i flowers, that is, the facial cycle of length $2i$ in Figure 4. Suppose $v \in V(C)$ is a leaf of T. In $G - v$, all vertices of C except one are cut vertices, so T may have at most one other vertex of C as a leaf. It follows that at most two vertices from C can be leaves in a spanning tree T of R_i. The remaining vertices of R_i that we have not considered yet (two for every flower) are cut vertices of R_i and therefore not leaves of T.

Summarizing, any spanning tree has at most two leaves in C, and at most four additional leaves for every flower. The statement follows. $\qquad \square$

2.2   Tightness of the Bound

The bound $n/3 + 4/3$ for cubic graphs is shown to be tight in [10]. Infinitely many examples are given with no more than $n/3 + 2$ leaves. On the other hand it is shown that there exists only one graph that ensures that the additive term $4/3$ can not be increased: the 3-dimensional cube Q_3, which has eight vertices and only admits four leaves.

Because the bound is best possible for cubic graphs, our bound is best possible as well. But also graphs with arbitrarily many vertices of higher and lower degree can be constructed which do not admit more than $n_{\geq 3}/3 + 2$ leaves. Figure 5 (a) shows such an example with many degree 2 and degree 4 vertices (which is closely related to one of the examples from [10]).

The reason that the additive term cannot be increased to 2 is again only one example: Figure 5 (b) shows a graph on $n = 7$ vertices that only admits $4 = n_{\geq 3}/3 + 5/3$ leaves. This graph will be called G_7 in the remainder. This graph is in fact a blossom plus two edges; deleting any edge between two degree 4 vertices yields a 2-blossom. An additive constant of $5/3$ is possible for non-cubic graphs, but we will not prove this statement in this version of the paper.
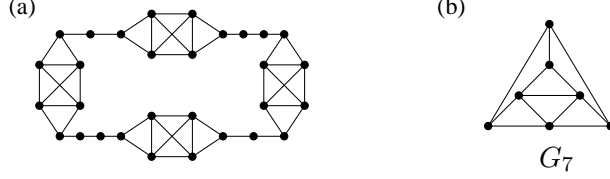

(a)   (b)

G_7

Figure 5: Non-cubic extremal graphs.

3 Proof of the Main Theorem

This section is devoted to the proof of the main theorem. We first sketch the proof and give an overview of the different ingredients that will be used. First we introduce a number of reduction rules in Section 3.1. These reduction rules are applied to the graph G until an irreducible graph G' is obtained. These rules have the property that if the main theorem holds for every component of G', it also holds for G. In the next sections, we therefore only have to consider irreducible graphs. In Section 3.2 we argue that the proofs from [10] for cubic graphs in fact show that if a non-spanning forest F of G' is given, that contains all vertices of G' of degree at least 4, then one of the trees of F can be extended to a larger tree while maintaining the proper leaf ratio. Finally, in Section 3.3 we show how to obtain this starting forest F that covers all high degree vertices, while having enough leaves. We use these tools in Section 3.4 to prove Theorem 1.

3.1 Reducible Structures

In this section we introduce a number of reduction rules. The proof of the main theorem relies on locally extending a forest until it becomes spanning while guaranteeing a certain number of leaves for every intermediate forest. The reductions help to delay the treatment of some substructures which cannot be readily handled during the extension process and they also simplify the case study in the main proof.

Ignoring rules that disconnect the graph, the main idea behind the reduction rules is as follows. A graph G is reduced to a graph G' with $n_{\geq 3}(G) - n_{\geq 3}(G') = k$, such that every spanning tree of G' can be turned into a spanning tree of G with at least $k/3$ additional leaves. This preserves the desired leaf ratio. Lemma 3 states this idea more precisely.

We now give the necessary definitions. A vertex with degree at most two will be called a *goober*. We adopt this notion from [10], although there it is defined differently. In [10] goobers are those vertices of degree at most two *resulting from a reduction rule*. We observe that nowhere in the proofs the extra structural information which this definition may provide is actually used. Hence goobers may simply be defined as we do here. The important gain is that now we do not have to require graphs to have minimum degree 3 in our statements.

goober

6

One important convention is that goobers are always defined with respect to the whole graph, that is when we consider a subgraph H of G, a vertex v of H is a goober if $d_G(v) \leq 2$. In our figures, white vertices indicate goobers. A *high degree vertex* is a vertex of degree at least 4.

We first repeat the seven reduction rules defined in [10], and then introduce five new rules which are designed to handle structures containing higher degree vertices. While the first seven rules are defined in [10] for graphs with maximum degree 3 we define them for arbitrary graphs, but the vertices on which they act must have the same degrees as in the original definition.

The seven reduction rules from [10] consist of graph operations on certain structures, and conditions on when they may be applied. Figure 6 shows the operations. The black vertices all have degree 3, and goobers are shown as white vertices. Dashed edges are present in the resulting graph if and only if they exist in the original graph. The numbers above the arrows indicate the decrease in $n_{\geq 3}$, and the numbers below the arrows indicate the number of leaves that can be gained in a spanning tree when reversing the reduction.
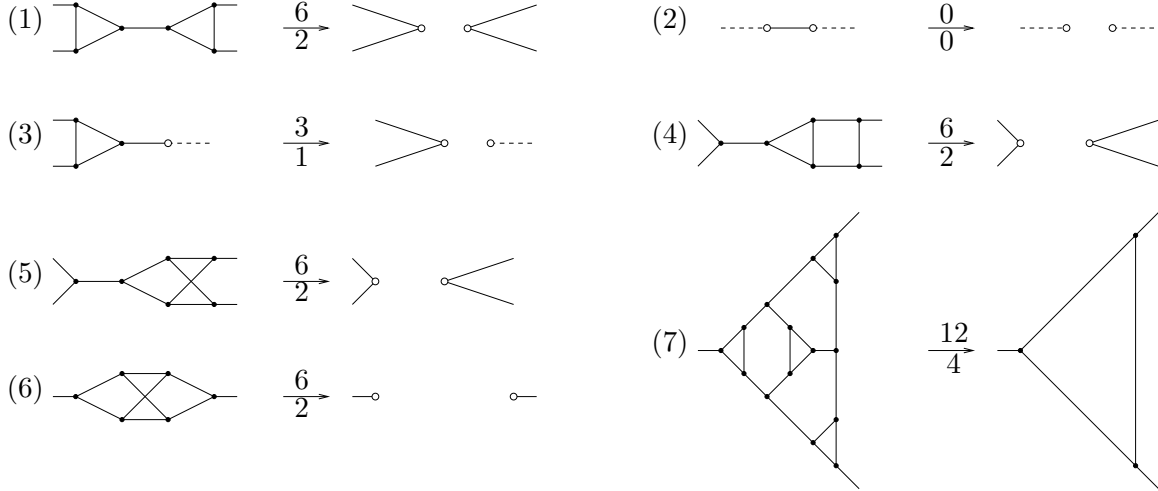

Figure 6: The seven low degree reduction rules

The following restrictions are imposed on the application of these rules (see Section 3 of [10]):

- Reductions (1), (3), (4) and (5) may not be applied if the two outgoing edges from the left side, or the two outgoing edges from the right side, share a non-goober end vertex. (An outgoing edge from the left and an outgoing edge from the right may share a non-goober end vertex.)

- Reduction (7) may not be applied if any pair of outgoing edges shares an end vertex.

In other words, a rule may not be applied if it would introduce multi-edges incident with non-goobers, or if it would introduce a diamond. These seven reduction rules will be called the *low-degree reduction rules*.

We define an invariant that exhibits the properties which should be maintained while doing graph reductions.

Definition 4 (Invariant). A graph H is said to *satisfy the invariant* if:

- H is connected, or every component of H contains a goober, and

7

- every component of H is either simple or it is a $K_2 + e$, and

- H contains neither 2-necklaces nor 2-blossoms.

The reduction rules are applied in the induction step in the proof of our main theorem; this invariant states the important properties that should be preserved in the reduction process.

Lemma 1. *Let G' be obtained from G by the application of a low-degree reduction rule. If G satisfies the invariant then so does G'.*

Proof. Note that the reduction rules (1)-(6) only introduce goobers as new vertices and the only new edges are incident to these goobers. Furthermore all other vertex degrees remain unchanged. Hence these reductions cannot introduce 2-necklaces or 2-blossoms. Rule (7) cannot introduce a 2-blossom since a 2-blossom cannot share a vertex with a triangle induced by three vertices of degree three. This is not true for 2-necklaces, but if rule (7) introduces a 2-necklace, two of the outgoing edges share an end vertex, contradicting the condition for applying rule (7).

For all of the rules that may disconnect the graph, it is clear that both new components will contain a goober. So the only way in which one of the reductions might violate the invariant is by introducing multiple edges. But using the imposed restrictions it can be seen that multiple edges can only be introduced between two goobers, giving a $K_2 + e$. □

We now introduce five new reduction rules, which we call the *high-degree reduction rules*. Each rule again consists of a graph operation and conditions on the applicability. Figure 7 shows the graph operations for the five rules.

The encircled vertices are the terminals, which may have further incidences, unlike the other vertices. None of the vertices in the figures may coincide, but there are no restrictions on outgoing edges sharing end vertices. The numbers above the arrows indicate the decrease in $n_{\geq 3}$, and the numbers below the arrows indicate the number of leaves that can be gained in a spanning tree when reversing the reduction. Since (R4) must disconnect a component, this notion is not relevant for (R4); this rule will be treated separately below.
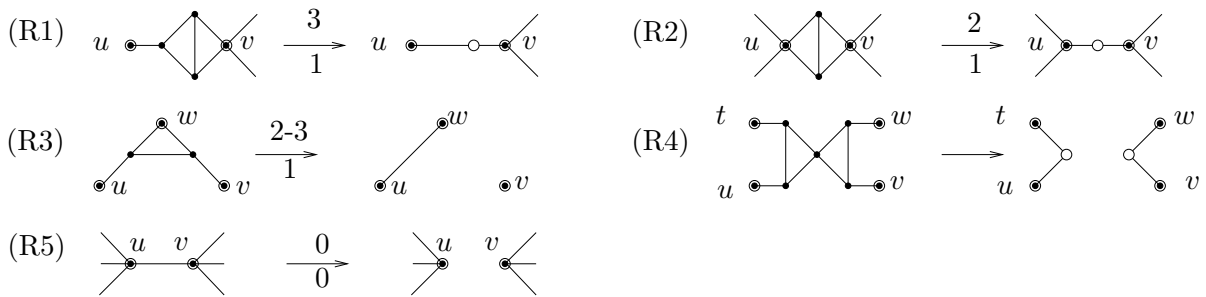

Figure 7: The high-degree reduction rules.

The following restrictions are imposed on the applicability of these operations to a graph G. First, none of the reduction rules may be applied if it introduces a new 2-necklaceor 2-blossom. In addition, the following rule-specific restrictions are imposed. Let $cc(H)$ denote the number of connected components of a graph H.

(R1) $d_G(v) \geq 4$.

(R2) $d_G(u) \geq 4$ and $d_G(v) \geq 4$.

(R3) $cc(G') = cc(G)$, the edge uw is not in G, and in addition $d_{G'}(v) \geq 3$, or $d_{G'}(w) \geq 3$, or both.

(R4) $cc(G') > cc(G)$, that is G' is *not* connected.

(R5) $d_G(u) \geq 4$, $d_G(v) \geq 4$, and uv may not be a bridge.

A *bridge* is an edge whose deletion increases the number of components. In the remainder, we will call a reduction rule *admissible* if it can be applied without violating one of the imposed conditions. In particular the condition that no 2-necklaces or 2-blossoms are introduced will be important. Since the high-degree reductions rules are defined such that no 2-necklaces or 2-blossoms can be introduced, it is easy to see that the following lemma holds: *bridge*   *admissible*

Lemma 2. *Let G' be obtained from G by the application of a high-degree reduction rule. If G satisfies the invariant then so does G'.*

Observe that in particular, (R5) seems counterproductive when the goal is to find spanning trees with many leaves, but it is useful to keep the case analysis in the proof of Lemma 6 simple.

Definition 5 (Reducible). A graph G is *reducible* if one of the low-degree or high-degree reduction rules can be applied, and *irreducible* otherwise.   *reducible*   *irreducible*

Griggs et al. [10] call a graph irreducible if none of the low-degree reduction rules can be applied. Clearly, a graph that is irreducible according to our definition is also irreducible according to their definition, so we may apply their lemmas for irreducible graphs also using the above definition of irreducibility.

Note that irreducible graphs satisfying the invariant are simple because of reduction rule (2). Components with only one vertex will be called *trivial components* in the sequel.   *trivial*   *components*

We now show that we can reverse all of these reduction rules while maintaining spanning trees for every component, having the proper number of leaves. For the low-degree reduction rules, this lemma was implicitly proved in [10]. So for the detailed tree reconstructions we refer to [10], but we do repeat the main idea behind the proof here.

Lemma 3 (Reconstruction Lemma). *Let G' be the result of applying a reduction rule to a connected graph G and $\alpha \geq 0$. If G' has k non-trivial components $C_1, \ldots, C_k$, which all have a spanning tree with at least $n_{\geq 3}(C_i)/3 + \alpha$ leaves, then G has a spanning tree T with*

$$\ell(T) \geq n_{\geq 3}(G)/3 + \alpha k - 2(k-1).$$

Note that the reduction rules create at most two components, that is $k \leq 2$. We use this lemma with $\alpha = 4/3$ if G' is connected, and with $\alpha = 2$ otherwise.

Proof. Suppose the applied rule was a low-degree reduction rule. Note that $cc(G')$ is either 1 or 2. If G' is connected, then its spanning tree can be turned into a spanning tree of G with $(n_{\geq 3}(G) - n_{\geq 3}(G'))/3$ more leaves. To prove this, it is shown in Section 3 of [10] for every rule how to adapt the tree of G' for G (*tree reconstructions*). This already proves the statement if $cc(G') = 1$. If $cc(G') = 2$, then applying the same tree reconstructions yields a spanning forest of G consisting of two trees, with again $(n_{\geq 3}(G) - n_{\geq 3}(G'))/3$ more leaves in

9

total. If both components of G' are non-trivial ($k = 2$), then the two resulting trees of G can be connected to one spanning tree T by adding one edge, losing at most two leaves. In that case we have:

$$\ell(T) \geq n_{\geq 3}(G')/3 + 2\alpha + (n_{\geq 3}(G) - n_{\geq 3}(G'))/3 - 2 = n_{\geq 3}(G)/3 + \alpha k - 2(k-1).$$

If exactly one of the two components is trivial ($k = 1$) then the applied rule must be Rule (2) or (3). In this case, it can be checked that after the tree reconstruction, one edge can be added without decreasing the number of leaves; one leaf is lost but an isolated vertex becomes a leaf. Then we have:

$$\ell(T) \geq n_{\geq 3}(G')/3 + \alpha + (n_{\geq 3}(G) - n_{\geq 3}(G'))/3 = n_{\geq 3}(G)/3 + \alpha k - 2(k-1).$$

If both components of G' are trivial ($k = 0$), then the applied rule was (2), and $G = K_2$, for which the statement holds: $-2(k-1) = 2$. This proves the lemma when a low-degree reduction rule is applied.
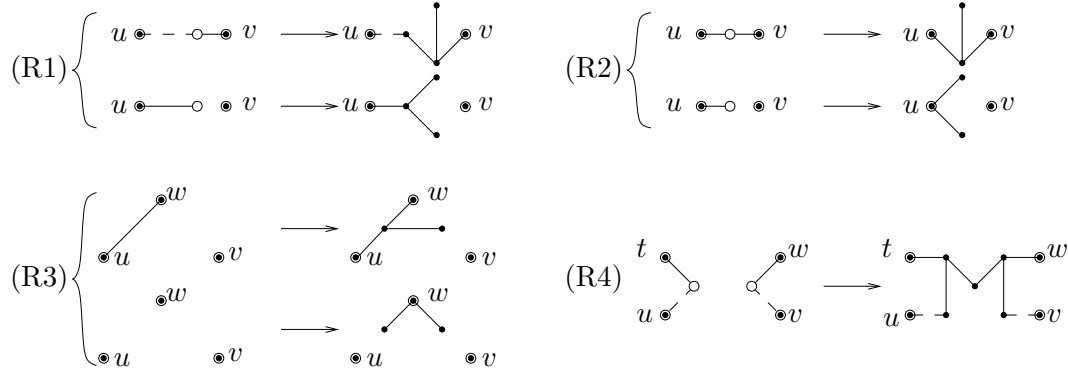

Figure 8: Spanning tree constructions when reversing the new reduction rules

Now we consider the high-degree reduction rules. Note that rules (R1), (R2), (R3) and (R5) do not increase the number of components, so $k = 1$. So for (R5) we do not have to change the spanning tree of G'. For (R1), (R2) and (R3), Figure 8 shows how to gain at least one additional leaf in every case, which suffices since each of these rules decreases $n_{\geq 3}$ by at most three. Here it is essential that (R3) is admissible only if it creates at most one goober. Dashed edges in the figure are present on the right if and only if they are present on the left. Symmetric cases are omitted in the figure. Note that none of the terminals of the operations can lose leaf status, except w in the second reconstruction for (R3). This is compensated by gaining two new leaves here. So in every case enough leaves are gained to maintain the ratio.

Recall that (R4) is only admissible if it disconnects G into two components, which will be non-trivial, so $k = 2$. Figure 8 shows how to construct a spanning tree for G from the two spanning trees for the components, without decreasing the total number of leaves. Hence the number of leaves of the resulting tree is at least

$$n_{\geq 3}(G')/3 + 2\alpha = n_{\geq 3}(G)/3 - 5/3 + 2\alpha > n_{\geq 3}(G)/3 + 2\alpha - 2 = n_{\geq 3}(G)/3 + \alpha k - 2(k-1)$$

This proves the lemma for all reduction rules. $\qquad\square$

The following property of irreducible graphs substantially simplifies subsequent proofs. Here G_7 denotes the graph from Figure 5 (b).

Lemma 4 (Edge Deletion). *Let G be an irreducible graph not equal to G_7 with adjacent vertices u and v. If $d(u) = d(v) = 4$, then uv is a bridge, or one of u, v becomes an inner vertex of a cubic diamond upon deletion of the edge uv.*

Proof. Suppose for the sake of contradiction a non-bridge edge uv exists, between vertices of degree 4, such that none of u, v becomes an inner vertex of a diamond upon deletion of uv.

Since G is irreducible, no reduction rule is admissible. Clearly, this must mean that a 2-necklace or 2-blossom is introduced when uv is deleted, that is when (R5) is applied to uv. In either case, we will derive a contradiction to the irreducibility of G.

Claim 1 The graph $G - uv$ does not contain a 2-necklace N.

Suppose for the sake of contradiction that $G - uv$ does contain a 2-necklace N. Consider N as a subgraph of G (so uv is counted towards the degrees of u and v).

We first treat the case that N consists of at least two diamonds. If one of the diamonds in N contains three vertices of degree 3, we can use rule (R1), see Figure 9 (a). So now we may assume that one diamond on the end of the necklace contains u as one of the three vertices not shared with the next diamond, and the diamond on the other end of the necklace contains v this way.

If u is a vertex with degree 2 in N, then (R2) can be applied, see Figure 9 (b). This does not introduce a 2-necklace since the degree 4 vertex v is part of N on the other end. Because v is part of a diamond, this can also not introduce a 2-blossom.

In the remaining case, both u and v are internal vertices of their respective diamonds. Now it is admissible to apply (R5) to a different edge incident with u, see Figure 9 (c), where the dashed edge is the deleted one.

This does not introduce a 2-necklace or 2-blossom: u becomes part of a triangle that is induced by degree 3 vertices, for which all outgoing edges have different end vertices. Such a triangle cannot be part of a 2-blossom or 2-necklace. The other end vertex of the deleted edge is still part of a diamond after deletion, and thus is not part of a 2-blossom. It is not part of a 2-necklace since v is in this part of the necklace.

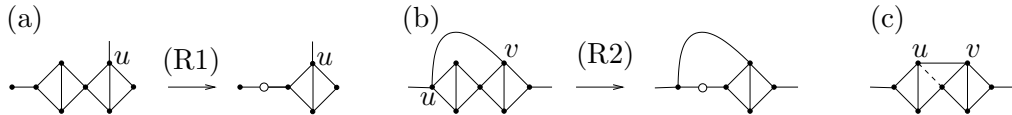This concludes the case where N consists of at least two diamonds.

Figure 9: Reductions when a long 2-necklace is created.

Now suppose N consists of a single diamond. If u is an inner vertex of this diamond, then v cannot be part of the same diamond since we are dealing with simple graphs. This is then the case we excluded by assumption, see Figure 10 (a). So without loss of generality u is one of the vertices that have degree 2 in the diamond.

Now rule (R1) or (R2) is admissible, depending on whether v is also in the diamond, see Figures 10 (b) and (c). This does not introduce a 2-blossom or 2-necklace, since in the case in Figure 10 (b), a triangle containing a goober is introduced, and in the case in Figure 10 (c), v has degree 4 and a goober at distance 2. Note that also no parallel edges are introduced: in the case in Figure 10 (c) the edges leaving the diamond are distinct, that is deleting uv does not give a K_4, since in that case uv would have been a bridge. This shows that it is admissible to apply either (R1) or (R2), which contradicts the irreducibility of G.   △
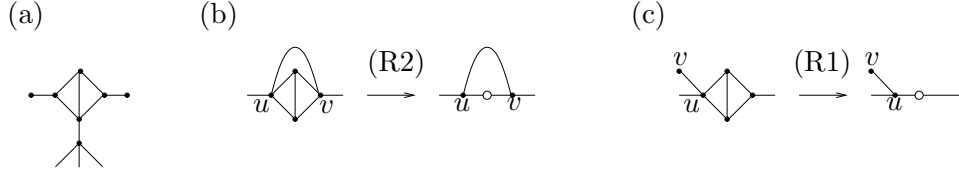
Figure 10: Reductions when a cubic diamond is created.

Claim 2 The graph $G - uv$ does not contain a 2-blossom B.

For B we use the vertex labels from Figure 11 (a). The degree 4 vertex of B is labeled b, its terminals are called c-vertices, and the remaining four vertices are called its a-vertices. Now consider B as a subgraph of G (so uv is counted towards the vertex degrees). Since $d_G(u) = 4 = d_G(v)$ neither of them is equal to b, since b has degree 4 even after the deletion of uv.
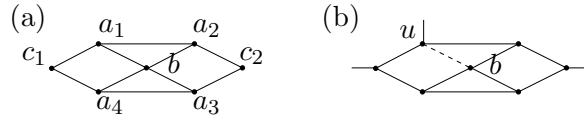

Figure 11: The blossom B after deleting uv.

If u is an a-vertex, say without loss of generality $u = a_1$, then it is admissible to delete the edge connecting u to b instead, see Figure 11 (b). We argue that this does not introduce a 2-blossom or 2-necklace. Figure 12 shows the possible results of deleting ub in more detail, depending on the position of v. First suppose $v \neq a_4$. After deleting ub, b becomes part of
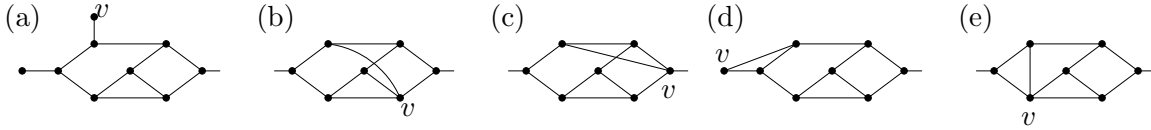

Figure 12: Possible results of deleting ub.

a triangle that does not share a vertex with another triangle, since we assumed $v \neq a_4$. It follows that b is neither part of a 2-necklace, nor of a 2-blossom. The vertex u may be part of a triangle (when $v = c_2$ or when v is not in B but adjacent to c_1), but such a triangle is not part of a diamond, hence u is not part of a 2-necklace. Finally we argue that u is not part of a 2-blossom: since b is not part of a 2-blossom, its neighbor a_2 is not part of a 2-blossom B' unless it is a terminal of B'. In that case it is not part of a triangle, but its neighbor c_2 is, which is impossible. Hence a_2 is not part of a 2-blossom. Then if u is part of a 2-blossom B', it must be a terminal of B', and thus not part of a triangle, but its neighbor c_1 must be part of a triangle. This is again not possible. This concludes the proof that if $v \neq a_4$, deleting ub is an admissible application of (R5).

Now we need to consider the case case that $v = a_4$ and $u = a_1$, see Figure 12 (e). Deleting ub does not introduce a 2-necklace, but there is exactly one way in which it may introduce a 2-blossom, which has v as its central degree 4 vertex. Figure 13 shows this case, the bold edges indicate the new blossom. But now it can be seen that the original graph, which includes ub,
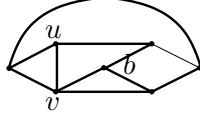
12

Figure 13: Deleting ub yields a blossom.

is exactly G_7, a contradiction with our assumption. We conclude that if u is an a-vertex and $G \neq G_7$, in every case the edge ub can be deleted by an admissible application of (R5).

It remains to consider the case that u is a c-vertex. Then, (R3) could be used, see Figure 14 (a) and (b). The bold edges indicate the structure reduced by (R3). If in case (a) a 2-necklace is introduced, v would be an inner vertex of one of its diamonds, but that is not possible since $d(v) = 4$. In case (b) no 2-necklace can be introduced, since v is part of at most one triangle. In neither case a 2-blossom is introduced. △
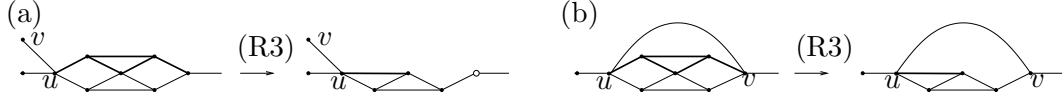

Figure 14: More reductions if a 2-blossom is created.

We have thus derived a contradiction to the irreducibility of the graph for all cases where deleting uv would not be an admissible application of (R5), which proves the lemma. □

3.2 Using the Result for Cubic Graphs to Prove Tree Extendibility

Using our observation that the results in [10] hold when goobers are simply defined as vertices with degree at most 2, we may restate Theorem 3 from [10] as follows.

Theorem 3. *Every irreducible graph G of maximum degree exactly 3 and without cubic diamonds has a spanning tree with at least $n_{\geq 3}(G)/3 + \alpha$ leaves, where $\alpha = 4/3$ if G is cubic and $\alpha = 2$ otherwise.*

We give a short overview of the proof of this statement, as it appears in [10]. For a subgraph T of G, in addition to $\ell(T)$ the following values are considered. By $n_G(T)$ we denote the number of non-goober vertices of G that are in $V(T)$. By $\ell_d(T)$ we denote the number of *dead leaves* of T, that is leaves of T, which have no neighbor in $V(G) \setminus V(T)$. $n_G(T)$ $\ell_d(T)$ *dead leaves*

The *value* of T is defined as $2.5\ell(T) + 0.5\ell_d(T) - n_G(T)$. First it is shown in [10] that a tree *value* T with value at least 4 can always be found, and even one with value at least 5.5 if H contains at least one goober. Next, it is shown that every non-spanning tree T can be *extended*, that is a tree supergraph T' of T can be found with value at least the value of T. This part of the proof consists of a rather involved case study. The extensions can be repeated until a spanning tree is found, in which case all leaves are dead. Rewriting the value expression, and rounding up the start value then yields Theorem 3.

We observe that nowhere in the case study that proves extendibility any information about the current tree T is used; loosely speaking, only information about the part of H 'outside' of T is used. In particular, the fact that T is connected is never used in the proof, and neither are upper bounds on degrees of vertices already included in T.

We will now define the *leaf potential* of a subgraph which generalizes the above definition of the value of a tree, and we will formalize the notions 'extendible' and 'outside'. Using these new notions a useful lemma can be formulated, which we conclude is proved, but not stated in [10].

Definition 6 (Leaf-Potential)**.** The *leaf-potential* of a subgraph $F \subseteq G$ is $\mathcal{P}_G(F) = 2.5\ell(F) + 0.5\ell_d(F) - n_G(F) - 6cc(F)$.

leaf-potential $\mathcal{P}_G(F)$

If ambiguities are excluded in the context we simply write $\mathcal{P}(F)$.

Definition 7 (Extendible)**.** Let F be a subgraph of a graph G. Then F is called *extendible* if there exists an F' with $F \subset F' \subseteq G$ and $\mathcal{P}_G(F') \geq \mathcal{P}_G(F)$.

extendible

Above we already informally mentioned the subgraph of G 'outside' a subgraph $F \subset G$. Considering the proof in [10], we see that this graph may formally be defined as an edge induced graph as follows.

Definition 8 (Graph Outside **F**)**.** Let F be a non-spanning subgraph of G. The *subgraph of G outside of F* is $F^C = G[\{uv \in E(G) : u \notin V(F)\}]$. The *boundary* of F is $V(F) \cap V(F^C)$

subgraph of G outside of F F^C boundary graph outside F

Note that no edges between two vertices that are both in $V(F)$ are included in F^C. If G is clear from the context we call F^C the *graph outside F*. Expressed using these definitions, the case study in [10] yields the following lemma.

Lemma 5 (Extension Lemma)**.** *Let G be a connected irreducible graph, and let $F \subset G$ such that F^C has maximum degree 3 and contains no cubic diamonds. Then F is extendible.*

3.3 Growing Trees around High Degree Vertices

The purpose of this section is to prove Lemma 6, which grows trees around high degree vertices and yields a graph satisfying the assumptions of Lemma 5. Lemma 6 is the core of our proof of Theorem 1.

We denote the set of vertices of G which are not in F by $\overline{V(F)} = V(G) \setminus V(F)$. The neighborhood $N(v)$ of a vertex v is the set of all vertices adjacent to v, and the closed neighborhood of v is $N[v] = N(v) \cup \{v\}$. *Expanding* a vertex $v \in V(G)$, which is an operation on a subgraph F of G, yields a new subgraph with vertex set $V(F) \cup N[v]$, and edge set $E(F) \cup \{uv : u \in N[v] \setminus V(F)\}$. So all newly added neighbors of v become leaves, and v may lose leaf status. The number of components increases by one if and only if $v \notin V(F)$. Expanding a list of vertices means expanding the vertices in the given order.

Expanding

We adopt the short-hand notation $\Delta(x, y, z) := 2.5y + 0.5z - x$ from [10] to express the change in $\mathcal{P}_G$ when extending a graph F to a new graph F'. Let $\Delta\ell$ denote $\ell(F') - \ell(F)$, and define $\Delta\ell_d$ and Δn_G analogously. So when F and F' have the same number of components, the extension is valid if and only if $\Delta(\Delta n_G, \Delta\ell, \Delta\ell_d) \geq 0$, and if a new component is introduced we need $\Delta(\Delta n_G, \Delta\ell, \Delta\ell_d) \geq 6$.

$\Delta(x, y, z)$

For the sake of simpler notation, instead of writing e.g. $\Delta(\Delta n_G, \Delta\ell, \Delta\ell_d) \geq \Delta(4, 3, 1) = 4$, we will simply write $\Delta(4, 3, 1) = 4$. Hence the three parameter values need not be exactly Δn_G, $\Delta\ell$ and $\Delta\ell_d$ but reflect the worst case scenario. That is, the change in the leaf potential that we prove is to be read as a lower bound for the actual change.

14

Lemma 6 (Start Lemma). *Let G be an irreducible graph not equal to G_7 and F a (possibly empty) subgraph of G, such that F^C contains at least one vertex of degree at least 4, and contains neither 2-necklaces or 2-blossoms. Then, F is extendible.*

Proof. First suppose F is not the empty graph. If there is a vertex v on the boundary of F which is not a leaf, then F' can be obtained by expanding v. There is no leaf lost since v was not a leaf, and the newly added vertices are leaves. So the augmentation inequality is satisfied: $\Delta(k, k, 0) \geq 0$. Hence, we may assume in the remainder that only leaves of F have neighbors in $\overline{V(F)}$, or in other words, all vertices on the boundary of F are leaves of F.

(A1) $\Delta(0,0,0) = 0$

(A2) $\Delta(i+1, i, 0) \geq 0.5$ if $i \geq 1$

(A3) $\Delta(1, 0, i) \geq 0$ if $i \geq 2$

(A4) $\Delta(i+2, i, j) \geq 0$ if $i, j \geq 1$

(A5) $\Delta(i+2, i, 0) \geq 1$ if $i \geq 2$

(A6) $\Delta(i+3, i, 0) \geq 0$ if $i \geq 2$
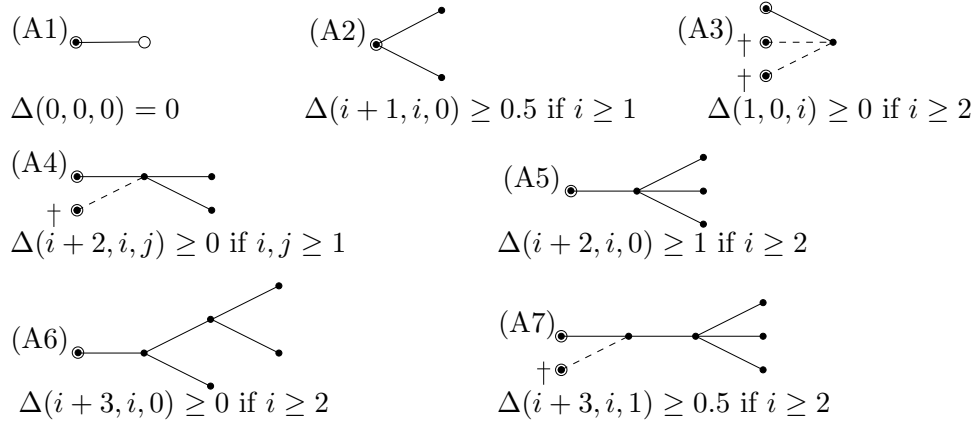
(A7) $\Delta(i+3, i, 1) \geq 0.5$ if $i \geq 2$

Figure 15: Simple augmentations of an existing subgraph

The next step is the attempt to augment F using the operations (A1)-(A7), see Figure 15. Conventions for this figure are that encircled vertices belong to $V(F)$, solid edges show the expansion and vertex degrees shown are to be understood as lower bounds. Dead leaves are marked with a cross. All of the expansions in the figure extend F without creating a new connected component, and satisfy $\Delta(\Delta n_G, \Delta \ell, \Delta \ell_d) \geq 0$. Thus the resulting graph F' is an extension as claimed in the lemma. Together these augmentation rules yield the following claim.

Claim 0 The subgraph F is extendible, if a vertex in $V(F)$ has a goober neighbor in $\overline{V(F)}$ or at least two neighbors in $\overline{V(F)}$, or if there is a high-degree vertex $v \in \overline{V(F)}$ at distance at most two from F.

If a goober from $\overline{V(F)}$ is adjacent to F then (A1) can be applied. If a vertex in $V(F)$ has at least two neighbors in $\overline{V(F)}$, (A2) can be applied. So from now on we will assume every vertex in $V(F)$ has at most one neighbor in $\overline{V(F)}$, and this neighbor is not a goober. If a high-degree vertex in $\overline{V(F)}$ is adjacent to a vertex in $V(F)$, (A3), (A4) or (A5) can be applied. The creation of the dead leaves in (A3) and (A4) follows from the fact that (A2) cannot be applied anymore. If a high-degree vertex in $\overline{V(F)}$ has distance two from a vertex in $V(F)$, (A6) or (A7) can be applied. $\triangle$

The rest of the proof will handle the more complicated case when F is the empty graph, or the only high-degree vertices in $\overline{V(F)}$ are at a larger distance from $V(F)$. We then introduce a new component for F. This is more complicated because adding a further component comes at a certain cost, more precisely we need that the new component satisfies $\Delta(\Delta n_G, \Delta \ell, \Delta \ell_d) \geq 6$.

The rest of the proof is divided into three more claims. The first one handles the easiest cases, and the second one handles all cases except those where every degree 4 vertex is the common vertex of two edge-disjoint triangles. This final case is then taken care of in the third claim. Throughout the proof we assume, sometimes implicitly, that none of the situations that have been handled earlier can occur.

Claim 1 Let $v \in \overline{V(F)}$, $d(v) \geq 4$, and $w \in N(v)$. In the following four situations F is extendible: $d(v) \geq 5$, or $d(v) = 4$ and w is a goober, or $d(v) = d(w) = 4$, or $d(v) = 4$ and $N[w] \subset N[v]$.

First note that no vertex in $N[v]$ or $N[w]$ is part of F by Claim 0. If $d(v) \geq 5$, expanding v yields $\Delta(k + 1, k, 0) \geq 6.5$, since $k \geq 5$. For $d(v) = 4$ and w a goober, expanding v gives $\Delta(4, 4, 0) = 6$.

Now suppose $d(v) = d(w) = 4$. If vw is a bridge, expanding v and w yields $\Delta(8, 6, 0) = 7$. Otherwise, Lemma 4 shows that either v or w, say v, becomes the inner vertex of a cubic diamond upon deletion of the edge vw. (Note that we assumed $G \neq G_7$, so Lemma 4 may be applied.) Thus, either w has two neighbors not in $N[v]$ and expanding v, w yields $\Delta(7, 5, 1) = 6$ (see Figure 16 (a)), or w shares two neighbors with v in which case expanding v yields $\Delta(5, 4, 3) = 6.5$ (see Figure 16 (b)).

So now we may assume that all neighbors of v have degree 3. If $N[w] \subset N[v]$ then either the unique vertex $u \in N[v] \backslash N[w]$ has two neighbors not in $N[v]$, in which case expanding u, v gives $\Delta(7, 5, 1) = 6$, see Figure 16 (c), or there is another vertex $x \in N[v] - w$ with $N[x] \subset N[v]$, and v is expanded to obtain $\Delta(5, 4, 2) = 6$, see Figure 16 (d). △
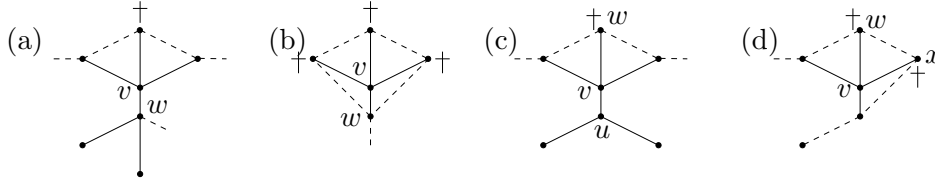

Figure 16: Figures for Claim 1.

Summarizing, we may now assume that $\overline{V(F)}$ contains no vertices of degree at least 5, and if it contains a vertex v of degree 4, all neighbors of v have degree 3 and have either one or two neighbors not in $N[v]$.

Claim 2 If $\overline{V(F)}$ contains a vertex v with $d(v) = 4$ and a vertex $w \in N(v)$ which has two neighbors $a, b \notin N[v]$, then F is extendible.

We denote the other three neighbors of v by x, y, z. If one of a, b, x, y, z has all of its neighbors in $\{a, b\} \cup N[v]$, we have $\Delta(7, 5, 1) = 6$ by expanding v, w, see Figure 17 (a).

If a or b is a goober we obtain $\Delta(6, 5, 0) \geq 6.5$, see Figure 17 (b). If a or b is adjacent to a vertex $c \in V(F)$, then expanding v, w will make c a dead leaf and yields $\Delta(7, 5, 1) \geq 6$, see Figure 17 (c). If one of a, b, x, y, z has at least two neighbors not in $N[v] \cup \{a, b\}$, we obtain $\Delta(9, 6, 0) = 6$ by expanding v, w and this vertex, see Figure 17 (d).

Hence we may assume that a, b, x, y, z each have exactly one neighbor outside $N[v] \cup \{a, b\}$. This neighbor is not part of F. Since they all have degree at least 3, these five vertices must induce three edges. This implies that one of a, b has degree 4 since we already know that
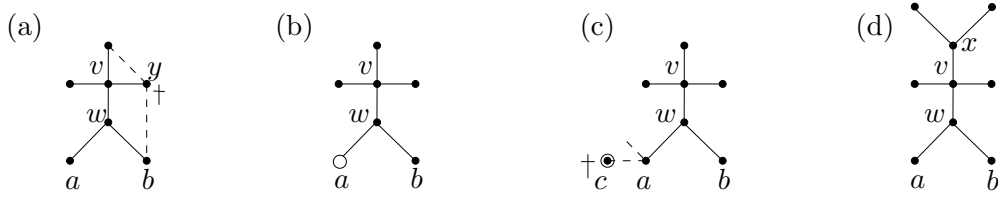
Figure 17: Figures for Claim 2.

x, y, z have degree 3. We may assume without loss of generality that $d(a) = 4$ and $d(b) = 3$. We distinguish two cases depending on whether a is adjacent to b or not. The statement 'a is adjacent to b' is denoted by $a \sim b$. We denote the neighbor of x outside of $N[v] \cup \{a, b\}$ by x', and similarly a', b', y', z' are defined. $a \sim b$

Case 1. a is adjacent to x and y while b is adjacent to z.

Consider expanding v, x, z. All vertices in $\{a, b, w, y, x', z'\}$ are adjacent to at least one of v, x, z, thus we have $\Delta(9, 6, 1) = 6.5$ unless $x' = z'$, see Figure 18 (a). By an analogous argument with y in the place of x we may now assume that $x' = z' = y'$. Then, expanding v, x yields $\Delta(7, 5, 1) = 6$, since y becomes a dead leaf, see Figure 18 (b).
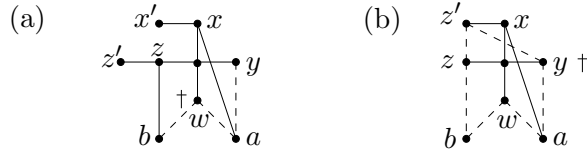

Figure 18: Figures for Claim 2, Case 1

Case 2. a is adjacent to b and x while y is adjacent to z.

If $x' \neq a'$, expanding a, x, v yields $\Delta(9, 6, 1) = 6.5$, see Figure 19 (a), so we may assume that $x' = a' =: c$, and this creates a situation symmetric in b and c. By Claim 1 we have that $b \not\sim c$. Now first suppose $b' \sim c$. Then expanding b', c, x, v yields $\Delta(10, 6, 3) = 6.5$, provided b' has a neighbor d other than y, z, see Figure 19 (b). Note that $d \in V(F)$ is not possible since augmentation (A6) could have been applied instead.
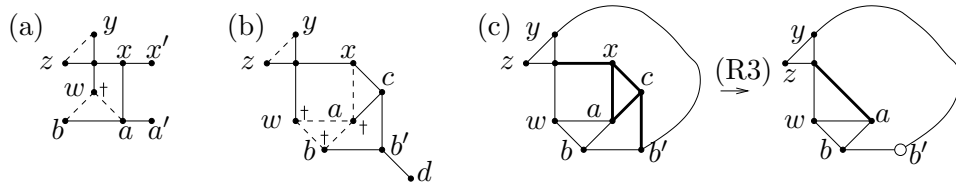

Figure 19: Figures for Claim 2, Case 2

If $N(b') = \{b, c, y\}$, then (R3) is admissible, see Figure 19 (c). Since b' becomes a goober this cannot introduce a 2-necklace. Hence it must be that $b' \sim y, z$ and the graph has $\Delta(9, 5, 5) = 6$, see Figure 20 (a). This concludes the cases with $b' \sim c$.

The case $b' \not\sim c$ can be excluded because then (R3) would be admissible, see Figure 20 (b). Note that this cannot create a 2-necklace involving y, z since then (R2) would have been admissible.
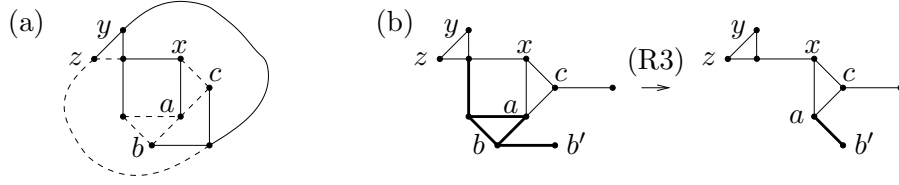
17

Figure 20: Additional figures for Claim 2, Case 2.

This concludes the proof of Claim 2. △

Summarizing Claims 0, 1, and 2, we may now assume that all neighbors of a degree 4 vertex $v \in \overline{V(F)}$ have degree 3, and have exactly one neighbor not in $N[v]$. In other words, v is the common vertex of two edge-disjoint triangles, see Figure 21.
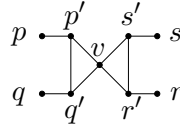

Figure 21: The bow tie subgraph

Claim 3 If the graph outside F contains a vertex v with $d(v) = 4$ such that all its neighbors have degree 3 and one neighbor outside $N[v]$, then F is extendible.

We denote the neighbors of v by p', q', r', s' and assume that $p' \sim q'$ and $r' \sim s'$. The neighbor of p' outside $N[v]$ is denoted by p and similarly q, r, s are defined, see Figure 21. We split the proof of the claim into three cases.

Case 1. $p = q$

If p has degree 3, we can apply (R1), see Figure 22 (a). So now without loss of generality p has degree 4. Then by Claim 2, p is also part of two edge-disjoint triangles. So if $p = r$ then also $p = s$. In that case we can expand p', v to obtain $\Delta(6, 4, 4) = 6$, see Figure 22 (b). So now $p \neq r$, $p \neq s$. Consider applying (R2) to the diamond consisting of p, p', q', v, see Figure 22 (c). If this introduces a 2-necklace, (R1) could have been applied to the diamond on the other end of this necklace. It cannot introduce a 2-blossom since the triangles of a 2-blossom contain a degree 4 vertex.
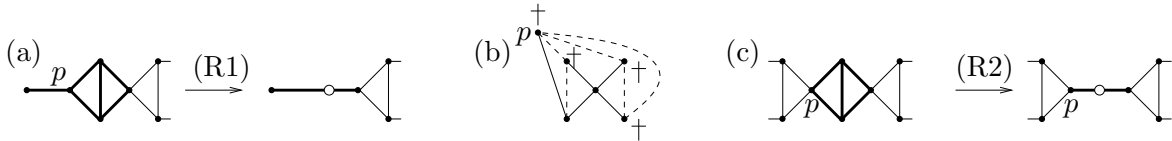

Figure 22: Figures for Claim 3, Case 1

Case 2. $p = r$

Note that $d(p) = 3$ by Claim 2. Also $q \neq s$ since the graph does not contain 2-blossoms and the case that $d(q = s) = 4$ is again excluded by Claim 2. Thus, (R3) is admissible, see Figure 23.
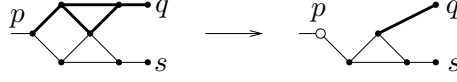
18

Figure 23: The figure for Claim 3, Case 2

Case 3. p, q, r, s pairwise different.

In this case either (R4) or (R3) is admissible: if v is a cut vertex, (R4) may be used (it increases the number of components). Otherwise, we may assume without loss of generality that p and s are in same connected component of $G - v$, and (R3) can be applied without disconnecting the graph. See Figures 24 (a) and (b). $\triangle$
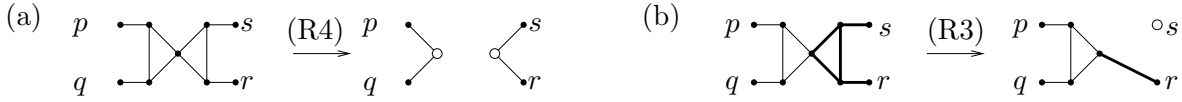

Figure 24: Figures for Claim 3, Case 3

This concludes all possible cases: whenever the subgraph of G outside of F contains a high degree vertex, we have shown that G is either reducible, or F is extendible. $\square$

3.4 The Proof of the Main Result

This section is devoted to combining the tools developed in the last three subsection in order to prove Theorem 1, which we repeat here for convenience.

Theorem 1. *Let G be a simple, connected graph on at least two vertices which contains neither 2-necklaces nor 2-blossoms. Then, G has a spanning tree T with*

$$\ell(T) \geq n_{\geq 3}(G)/3 + \begin{cases} 4/3 & \text{if } \delta(G) \geq 3 \\ 2 & \text{if } \delta(G) \leq 2. \end{cases}$$

Proof. We prove the statement by induction. For our induction hypothesis we actually prove that the above statement holds for every connected graph which satisfies the invariant. Then the statement follows for simple graphs.

First suppose G is irreducible. If G has maximum degree exactly 3, Theorem 1 follows immediately from Theorem 3. If G has maximum degree at most 2, G has a spanning tree with at least two leaves (note that we assumed that G is not a K_1), which suffices. If $G = G_7$, then a spanning tree with $4 = n_{\geq 3}(G)/3 + 5/3$ leaves can be obtained. So we may now assume that G contains at least one high degree vertex, and is not equal to G_7.

We start with an empty subgraph F of G, which has $\mathcal{P}_G(F) = 0$. The Start Lemma (Lemma 6) shows that, as long as there is at least one high degree vertex not in F, we can extend F while maintaining $\mathcal{P}_G(F) \geq 0$. When all high degree vertices are included in F, the Extension Lemma (Lemma 5) can be applied iteratively, until a spanning subgraph F' is obtained with $\mathcal{P}_G(F') \geq 0$. Without loss of generality, we may assume that F' is a forest; cycles can be broken without decreasing the number of leaves. Since all leaves of a spanning subgraph are dead we deduce

$$0 \leq \mathcal{P}_G(F') = 3\ell(F') - n_{\geq 3}(G) - 6cc(F') \implies \ell(F') \geq n_{\geq 3}(G)/3 + 2cc(F').$$

19

We can now add $cc(F') - 1$ edges to F' to obtain a spanning tree, losing at most $2(cc(F') - 1)$ leaves, so the resulting tree has at least $n_{\geq 3}(G)/3 + 2$ leaves.

It remains to consider the case that G is reducible (the induction step). Some reduction rule is admissible, and the reduced graph G' again satisfies the invariant, by Lemmas 1 and 2.

First suppose G' is connected. None of the reduction rules remove goobers, so if $\delta(G) \leq 2$, then $\delta(G') \leq 2$, and by induction G' has a spanning tree with at least $n_{\geq 3}(G')/3 + 2$ leaves. Lemma 3 then shows that G has a spanning tree with at least $n_{\geq 3}(G)/3 + 2$ leaves. Similarly, if $\delta(G) \geq 3$ then it follows that G has a spanning tree with at least $n_{\geq 3}(G)/3 + 4/3$ leaves. Now suppose the reduction rule yields a disconnected graph G'. Then, by the definition of the reduction rules, every resulting component has a goober. So by induction, every non-trivial component C of G' has a spanning tree with at least $n_{\geq 3}(C)/3 + 2$ leaves. Thus Lemma 3 implies that G has a spanning tree with at least $n_{\geq 3}(G)/3 + 2$ leaves. □

4   A fast FPT Algorithm for MaxLeaf

In this section we present a fast and relatively simple FPT algorithm for MaxLeaf, which uses Theorem 1 as an essential ingredient. The other two ingredients are a short preprocessing step, consisting of two reduction rules, and an enumerative procedure, which is similar to the one introduced in [3], and also applied in [2].

We start by presenting the two reduction rules that constitute the preprocessing phase. The reduction rules for the FPT algorithm are different from the rules used in Section 3. It is important that they yield an equivalent instance of the decision problem MaxLeaf, but there are no conditions on the ratio between the decrease in vertices and in possible leaves.

Recall that in 2-necklaces and 2-blossoms both terminals have degree 3 in G. The rules we introduce now also reduce diamonds and blossoms whose two terminals have arbitrary degree. However the two terminals of the subgraph must still be the two vertices that have degree 2 in the diamond necklace or blossom itself. Such a subgraph of G will be called a *2-terminal diamond* respectively a *2-terminal blossom*. Rule (F1) in Figure 25, which resembles rule (R2), reduces 2-terminal diamonds. Since the 2-necklace N_k consists of k 2-terminal diamonds it is reduced as well by rule (F1). Rule (F2) in Figure 25 reduces 2-terminal blossoms. The next lemma proves the correctness of these rules.

2-terminal diamond
2-terminal blossom

Figure 25: Two reduction rules for an instance (G, k) of MaxLeaf.

Lemma 7. *Let G' be the result of applying reduction (F1) or (F2) to G. Then $(G', k - 1)$ is a YES-instance for MaxLeaf if and only if (G, k) is a YES-instance for MaxLeaf.*

Proof. First consider the case where the applied rule was (F1), reducing a 2-terminal diamond D of G with terminals u and v. Consider a spanning tree T of G with at least k leaves. Observe that in T, we can always replace the set of edges $E(T) \cap E(D)$ by one of the two sets shown on the right in Figure 26 (a), or a symmetric set, without decreasing the number of leaves. Then by replacing it by the corresponding structure on the left, we obtain a spanning tree T' of G' with at least $k - 1$ leaves. Note here that the terminals u and v remain leaves in

T' if they are leaves in T. Similarly, any spanning tree of G' has one of the forms shown on the left when restricted to the two edges resulting from the reduction. Replacing it with the corresponding structure on the right shows that if $(G', k-1)$ is a YES-instance, (G, k) is as well. Figure 26 (b) can be used to prove the statement for rule (F2) analogously. For this it is useful to note that Proposition 1 shows that we may assume that T restricted to the blossom has one of the forms on the right of Figure 26 (b). $\square$
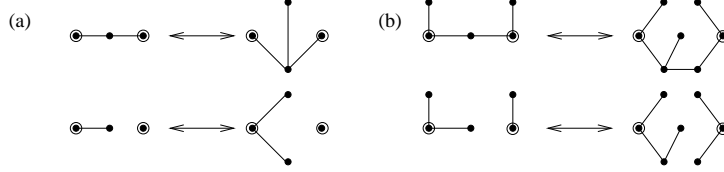

Figure 26: Tree reconstructions for (F1) and (F2).

Throughout this section we will denote the set of leaves of a graph G by $L(G)$. We now explain how to obtain a graph $S(G)$ from a graph G by suppressing vertices. *Suppressing* a vertex u of degree 2 means deleting u and adding an edge between the two neighbors of u, or a loop if both edges incident with u end in the same vertex. We allow this operation to introduce parallel edges and loops, so the degrees of non-suppressed vertices are maintained. If $|V_{\geq 3}(G)| = 0$, that is G is a path or cycle, then $S(G)$ is the empty graph. If $|V_{\geq 3}(G)| > 0$ then $S(G)$ is obtained from G by suppressing all degree 2 vertices. So $V(S) = L(G) \cup V_{\geq 3}(G)$, and G is a subdivision of $S(G)$. Hence loops and non-loop edges of $S(G)$ correspond to cycles and paths of G respectively. Let uv be a non-loop edge of $S(G)$ where the corresponding path P_{uv} in G has i internal vertices. We define a cost function c on the non-loop edges of $S(G)$ which assigns cost $c(uv) = \min\{i, 2\}$ to uv. Thus $c(uv)$ is the maximum possible number of leaves that a spanning tree of G can have among the internal vertices of P_{uv}. Now we are ready to present the FPT algorithm in Algorithm 1.

$L(G)$
$S(G)$
Suppressing

$c(uv)$

Algorithm 1 An FPT algorithm for MaxLeaf

INPUT: a MaxLeaf instance (G, k).

1) **while** G has a 2-terminal diamond or 2-terminal blossom subgraph **do**
    $G :=$ the result of applying (F1) or (F2) to G
    $k := k - 1$
  end while
2) **if** $n_{\geq 3}(G) \geq 3k$ or $|L(G)| \geq k$ or $k \leq 2$ **then** return(YES) **endif**
3) construct $S(G)$ and c
4) **for** all $L \subseteq V_{\geq 3}(G)$ with $|L| \leq k$ **do**
    if G has a spanning tree T with $L \subseteq L(T)$ and $|L| + |L(T) \backslash V_{\geq 3}(G)| \geq k$ **then**
      return(YES)
    endif
  endfor
5) return(NO)

In the following proofs, we will use the fact that for any $L \subset V(G)$, a spanning tree T of G with $L \subseteq L(T)$ exists if and only if $G - L$ is connected and $V(G) \backslash L$ is a dominating set.

The decision in Step 4 can be made in polynomial time in the size of $S(G)$. The essential step is to solve a minimum weight spanning tree problem on $S(G) - L$, using edge costs c. Lemma 8 contains the details.

Lemma 8. *Let (G, k) be a* MaxLeaf *instance for which $S(G)$ and c are non-empty and known. For any $L \subseteq V_{\geq 3}(G)$, deciding whether G has a spanning tree T with $L \subseteq L(T)$ and $|L| + |L(T) \backslash V_{\geq 3}(G)| \geq k$ can be done in time polynomial in the size of $S(G)$.*

Proof. Let $S = S(G)$. A spanning tree T of G with $L \subseteq L(T)$ exists if and only if $V(G) \backslash L$ is a connected, dominating set of G. This is the case if and only if $V(S) \backslash L$ is a connected, dominating set of S and there is no edge $uv \in E(S)$ with $u, v \in L$ and $c(uv) \geq 1$. These properties can be checked in time polynomial in the size of S.

Now suppose at least one spanning tree T_0 of G with $L(T_0) \subseteq L$ exists. We show how to construct such a spanning tree T that maximizes $|L(T) \backslash V_{\geq 3}(G)|$. This process is illustrated in Figure 27. White vertices indicate vertices in L.
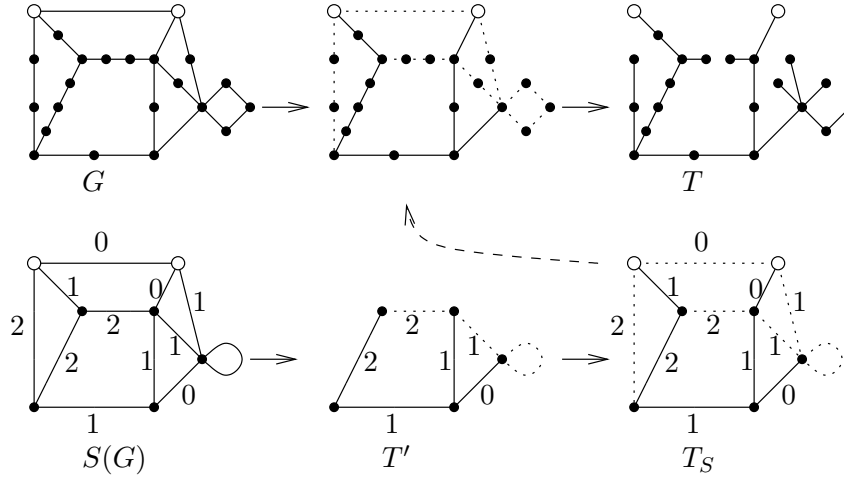

Figure 27: Constructing a tree T with $L \subseteq L(T)$ using $S(G)$.

First let T' be a minimum weight spanning tree of $S - L$, with respect to the cost function c. (This tree can be found in polynomial time.) A spanning tree T_S of S is obtained from T' by connecting every $u \in L$ to T' by an edge e which has minimum cost $c(e)$. So $L \subseteq L(T_S)$.

A spanning tree T of the subdivision G of S can be obtained from T_S in the following way. All edges from paths of G that correspond to edges in T_S are in T. At this stage T need not be spanning. The inner vertices of a path P_{uv} of G corresponding to an edge $uv \in E(S) \backslash E(T_S)$ with $u \in L, v \notin L$ are connected to T such that u remains a leaf. The inner vertices of a path P_{uv} with $uv \in E(S) \backslash E(T_S)$ and $u, v \notin L$ can be connected to T such that $c(uv)$ of them become leaves of T. Finally, vertices of cycles of G that correspond to loops of S are connected to T such that two of them become leaves. Note that an edge $uv \in E(S) \backslash E(T_S)$ with $u, v \in L$ must correspond to a path P_{uv} in G with no inner vertex since $G - L$ is connected. At this point, every vertex of G is connected to T, without introducing cycles, hence T is a spanning tree.

It can be verified that T was constructed such that $L(T) \backslash V_{\geq 3}(G)$ is maximized, under the condition that $L \subseteq L(T)$. For this it is essential that T' was chosen to be a minimal spanning

tree of $S - L$. We omit the formal proof of this fact here, noting that it is similar to the proof given in [3] and [2].

Observe that T does not actually have to be constructed for the decision. Thus only $S(G)$ needs to be considered, and the statement follows. $\qquad \square$

Lemma 9. *Algorithm 1 returns YES if and only if its input (G, k) is a YES-instance for* MaxLeaf.

Proof. Lemma 7 shows that it suffices to prove the statement for the reduced instance (G, k). Note that the reduced instance (G, k) is again simple, connected and non-trivial and does not contain 2-terminal diamonds or 2-terminal blossoms and thus also no 2-necklaces or 2-blossoms. So if $n_{\geq 3}(G) \geq 3k$, then (G, k) is a YES-instance by Theorem 1. The correctness of the other cases in which the algorithm returns YES is easily checked.

Suppose that (G, k) is a YES-instance. We show that the algorithm indeed returns YES. If $V_{\geq 3}(G) = \emptyset$, then G is a path or cycle, so (G, k) being a YES-instance implies $k \leq 2$, and YES is returned in Step 2. Suppose $V_{\geq 3}(G) \neq \emptyset$, Step 2 did not return YES, and T is a spanning tree of G with at least k leaves. If $|L(T) \cap V_{\geq 3}(G)| \geq k$, then some set $L \subseteq L(T) \cap V_{\geq 3}(G)$ with $|L| = k$ is considered in the algorithm. Clearly also a spanning tree T' exists with $L \subseteq L(T')$, so the algorithm returns YES in Step 4. On the other hand, if $L = L(T) \cap V_{\geq 3}(G)$ has fewer than k elements, then L itself will be considered in Algorithm 1, and in addition $|L| + |L(T) \backslash V_{\geq 3}(G)| = |L(T)| \geq k$. The algorithm will then return YES in Step 4. $\qquad \square$

Lemma 9 proves the correctness of Algorithm 1, the claimed time complexity will be proved next. Together this proves Theorem 2. We repeat the statement for convenience.

Theorem 2. *There exists an FPT algorithm for* MaxLeaf *with time complexity $O(m) + O^*(6.75^k)$, where m denotes the size of the input graph and k the desired number of leaves.*

Proof. It only remains to prove the complexity bound.

The first three steps can be done in linear time by building the proper data structures. For this it is essential that the degree of non-terminal vertices of 2-terminal diamonds and blossoms is bounded by a constant. We give more details now.

Assume that G is represented by doubly linked adjacency lists. It is possible to detect 2-terminal diamonds and 2-terminal blossoms because all of their non-terminal vertices have degree at most 4. For every vertex u of degree at most 4, we can store the position of u in the adjacency lists of its neighbors. To do this for every such u, all adjacency lists of the graph need to be scanned only once. This information now makes it possible to apply (F1) and (F2) in constant time for each diamond and blossom respectively.

Step 2 can obviously be done in linear time. For Step 3 we switch to a representation using arrays for the edges, which contain the labels of the two end vertices and the edge weights. We also store the vertex degrees, and the labels of the incident edges for vertices of degree 2. This representation allows us to do the following operations in constant time: suppressing a degree 2 vertex, calculating the resulting edge weight, and updating the representation.

Thus Steps 1-3 can be performed in linear time and it only remains to consider the complexity of Step 4. Since the reductions in Step 1 do not increase the number of vertices or the value of k, we may assume that n and k are the number of vertices and the parameter of the reduced instance, as it is after Step 1.

Step 4 of the algorithm is only executed when $|V_{\geq 3}(G)| < 3k$ and $|L(G)| < k$. Furthermore $V(S(G)) = L(G) \cup V_{\geq 3}(G)$, so every iteration of the for-loop of Step 4 takes time polynomial

in k (Lemma 8). This for-loop is executed once for every subset $L \subseteq V_{\geq 3}(G)$ with $|L| \leq k$. Using $|V_{\geq 3}(G)| \leq 3k$, the number of such sets can be verified to be $O(k\binom{3k}{k})$. Using Stirling's approximation $x! \approx x^x e^{-x} \sqrt{2\pi x}$, we obtain

$$\binom{3k}{k} = \frac{(3k)!}{(2k)!k!} \in O\left(\frac{(3k)^{3k}}{e^{3k}} \cdot \frac{e^{2k}}{(2k)^{2k}} \cdot \frac{e^k}{k^k}\right) = O\left(\frac{3^{3k}}{2^{2k}}\right) = O(6.75^k).$$

This concludes the proof. $\qquad\square$

We remark that we did not optimize the polynomial factor suppressed by the O^* notation, but it can be seen to be a practical, low degree polynomial.

5   Conclusions

We conclude with some remarks about possible improvements. Theorem 1 can be strengthened at the cost of lengthier proofs. An extended version of this paper will show that $n_{\geq 3}(G)/3 + 2$ leaves can be obtained whenever G is not cubic, and not equal to G_7. It can also be shown that in order to obtain this bound, 2-blossoms do not have to be excluded; it suffices to only exclude larger structures like the flower in Figure 2. This way a bound of $4n_{\geq 3}(G)/13 + c$ can be proved when only 2-necklaces are excluded.

Besides optimizing the parameter function of FPT algorithms, another goal is to find better *kernelizations*. For MaxLeaf, a kernelization can be defined as a preprocessing method that reduces the input to an instance (G, k) which is either a YES-instance, or has $|V(G)| \leq f(k)$ for some function $f(k)$. The current best kernelization for MaxLeaf has $f(k) = 3.75k$, see [6]. Since our goal was to avoid preprocessing as much as possible, our method does not give a kernelization by itself. But for instance our approach can be combined with three simple reduction rules from [3] and [6] to remove all leaves and adjacent degree 2 vertices. This yields a $7k$ kernelization: a reduced NO-instance has less than $3k$ vertices of degree at least 3 by Theorem 1, and less than $4k$ vertices of degree 2.

References

[1] H. L. Bodlaender. On linear time minor tests with depth-first search. *J. Algorithms*, 14(1):1–23, 1993.

[2] P. S. Bonsma. *Sparse cuts, matching-cuts and leafy trees in graphs.* PhD thesis, University of Twente, Enschede, the Netherlands, 2006.

[3] P. S. Bonsma, T. Brueggemann, and G. J. Woeginger. A faster FPT algorithm for finding spanning trees with many leaves. In *MFCS 2003*, volume 2747 of *LNCS*, pages 259–268. Springer, Berlin, 2003.

[4] R. G. Downey and M. R. Fellows. Parameterized computational feasibility. In *Feasible mathematics, II (1992)*, volume 13 of *Progr. Comput. Sci. Appl. Logic*, pages 219–244. Birkhäuser Boston, 1995.

[5] R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer-Verlag, New York, 1999.

[6] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond. FPT is P-time extremal structure I. In *ACiD 2005*, volume 4 of *Texts in algorithmics*, pages 1–41. King's College Publications, 2005.

[7] M. R. Fellows, C. McCartin, F. A. Rosamond, and U. Stege. Coordinatized kernels and catalytic reductions: an improved FPT algorithm for max leaf spanning tree and other problems. In *FST TCS 2000*, volume 1974 of *LNCS*, pages 240–251. Springer, Berlin, 2000.

[8] J. Flum and M. Grohe. *Parameterized complexity theory*. Springer, Berlin, 2006.

[9] M. R. Garey and D. S. Johnson. *Computers and intractability*. Freeman, San Francisco, 1979.

[10] J. R. Griggs, D. J. Kleitman, and A. Shastri. Spanning trees with many leaves in cubic graphs. *J. Graph Theory*, 13(6):669–695, 1989.

[11] D. J. Kleitman and D. B. West. Spanning trees with many leaves. *SIAM J. Discrete Math.*, 4(1):99–106, 1991.

[12] N. Linial and D. G. Sturtevant. Unpublished result, 1987.